Data Transformation and Semantic Log Purging for Process Mining*

Linh Thao Ly[1], Conrad Indiono[2], Jürgen Mangler[2], and Stefanie Rinderle-Ma[2]

Institute of Databases and Information Systems, Ulm University, Germany
thao.ly@uni-ulm.de
Faculty of Computer Science, University of Vienna, Austria
{conrad.indiono, juergen.mangler, stefanie.rinderle-ma}@univie.ac.at

Abstract. Existing process mining approaches are able to tolerate a certain degree of noise in process log. However, processes that contain infrequent paths, multiple (nested) parallel branches, or have been changed in an ad-hoc manner, still pose challenges. For such cases, process mining typically returns "spaghetti-models", that are hardly usable even as a starting point for process (re-)design. In this paper, we address these challenges by introducing data transformation and pre-processing steps that improve and ensure the quality of mined models for existing process mining approaches. We propose the concept of semantic log purging, i.e., the cleaning of logs based on domain specific constraints utilizing knowledge that typically complements processes. Furthermore we demonstrate the feasibility and effectiveness of the approach based on a case study in the higher education domain. We think that semantic log purging will enable process mining to yield better results, thus giving process (re-)designers a valuable tool.

Key words: Process mining, Data transformation, Log purging, Constraints

1 Introduction

Process Mining has developed as promising technique for analyzing process-oriented data in various ways: process discovery refers to the extraction and establishment of process models, process analysis addresses the analysis and comparison of processes. Both kinds of techniques are based on logs that store temporally ordered events about process executions. Though the Process Mining Manifesto [1] states that process event logs should be treated as "first class citizens" by enterprises, reality is often different: data are distributed over several sources and are often not directly available as temporally structured event logs. However, neglecting such "second class level" data sources might lead to missing out relevant and valuable analysis results as well as limit the applicability of process mining techniques dramatically. Hence, in order to utilize application

* The work presented in this paper has been partly conducted within the project I743 funded by the Austrian Science Fund (FWF).

data sources, the development of adequate methods for *data transformation* as pre-processing phase of a process mining project becomes essential.

Similar to data mining, data quality is crucial for process mining. In this case, the heterogeneity of log data sources (including applications that enable non-sequential or flexible execution of tasks) might account for the following cases that usually hamper the application of process mining techniques:

1. Incorrect/incomplete log data (in the following addressed as noise).
2. Log data contributed by parallel branches.
3. Infrequent traces.
4. Log data contributed by ad-hoc changed instances.

Case **(1)** has been mainly tackled at the algorithmic level. The *Genetic Miner* [2] and the *Heuristics Miner* [3], for example, are by design able to tolerate a certain degree of noise in the logs. We do not focus on such kinds of incorrect / incomplete log data and rely on the features of the process mining tool. For case **(2)**, post-processing methods in order to reduce the "spaghetti-degree" of the resulting process models have been proposed [4]. A special case worth mentioning is processes which include late modeling (e.g. BPMN ad-hoc subprocesses). Allowing to invoke a set of optional process steps in arbitrary order is identical to the modeling of parallel branches for all possible sequence variations of these steps. Detection of such cases is, to the best of our knowledge, currently not possible. **(3)** Infrequent traces are the result of normal process execution, but sparsely occur in process logs because they depend on rare conditions (e.g. credit lines over €100 million have to be signed by the board of directors). Sometimes they are implemented using automatic exception handling logic as can be defined in current process execution languages (i.e. planned exception handling). Current frequency-based approaches would discard cases of category **(3)**. Case **(4)**, to the best of our knowledge, is currently not covered by existing approaches: results may range from wrong process models to complete graphs (for parallel or ad-hoc changed instances). Ad-hoc changed instances typically involve the intervention of humans (i.e. unplanned exception handling).

In this paper, we will address the following two research questions:

1. *Data transformation:* How can we support the (automatic) transformation of temporal application data to process logs?
2. *Data quality:* Is it possible to increase the process mining quality for existing mining algorithms through semantic pre-processing?

The contribution of this paper is two-fold. First, we will present a method for query-based data collection and transformation of temporal application data into process logs. Secondly, a data cleaning method based on *semantic log purging* will be proposed. This method utilizes semantic knowledge on the processes in the form of process constraints. Based on the process constraints, (1) process logs

can be checked for instances that violate one or more process constraints. (2) The log can be filtered according to expected behavior thus allowing to untangle ad-hoc or highly parallel sections. In both cases, filtering out constraint violations can lead to an improvement of the process mining results in terms of reduced "spaghetti-degree" [5] of the processes.

In order to evaluate our approaches on data transformation and semantic log-purging for data cleaning, we apply them to a realistic data set from the higher education domain (HEP project, `www.wst.univie.ac.at/communities/hep/`). The HEP data set consists of ten different process types (reflecting different courses). For this paper we selected one course, which took place in three consecutive years. Collecting data for this course yielded 330 instances (students) with 18511 events. We also, together with the instructors, collected a set of process constraints, that served as the basis for the sematic log purging. In the end we used a goal process to compare the result quality of the mined processes.

The paper is structured as follows: In Sect. 2, we introduce the applied methodology and provide fundamentals on the HEP data set. Sect. 3 addresses the transformation of temporal application data to process logs. Semantic log purging is introduced in Sect. 4. The results from our study are summarized in Sect. 5. Related work is discussed in Sect. 6. The paper closes with a summary and an outlook in Sect. 7.

2 Applied Methodology and the HEP Data Set

As mentioned in the introduction, this paper focuses on new techniques for data transformation and cleaning in order to improve the quality of mined processes, embedded within a case study to show the feasibility of the approach. In this section, we describe considerations regarding the applied methodology as well as the raw data set and the associated reference process models.

2.1 Methodology

Figure 1 depicts the methodology applied in this paper, comparing process mining results with and without pre-processing by semantic log purging.

Data transformation Traditionally, the raw log data is transformed into some kind of data format. In the case of applications that aggregate several process steps, this may also include transformation and import of data into a database.

Process mining In order to obtain a process model from the logs, we apply the *Heuristics Miner* using the ProM process mining framework [6]. Being the mining approach most resilient towards noise [3], the Heuristics Miner is not as prone to deriving "spaghetti models" as for example the *Alpha Algorithm*. All mining algorithms in general seem to have problems with parallel executions which are seldom correctly detected and lead to a cobweb of intertwined tasks.

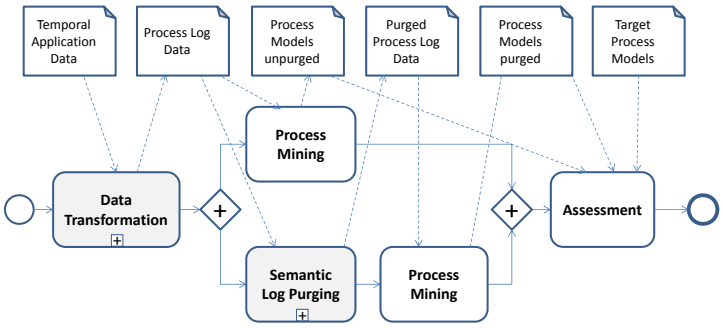

Fig. 1. Methodology for Evaluating Effectiveness of Semantic Log Purging

Assessment In the final step, models are typically analyzed and sanity-checked.

As outlined in the introduction, our goal is to clean the log in order to improve the overall quality of the mined process models. We assume that the quality of a mined process is high when an assessment concludes that: (i) Infrequent traces are correctly included in the result. (ii) Parallel branches are correctly identified. Logs stemming from processes which include late modeling yield meaningful results (a small set of branches). (iii) Ad-hoc changed instances stemming from manual repair are not incorporated in the process model. This implies that no tasks and/or edges have to be removed or added.

Our idea is that with a small set of constraints we are able to (1) filter the logs on a semantic basis without purging infrequent traces, (2) separate parallel branches, and (3) select typical execution patterns for ad-hoc execution. Thus we introduced the lower branch in Fig. 1 where we redefine/add the steps:

Data transformation In order to allow for a wider range of data sources, we introduce a DSL (domain specific language) that allows for performant, stream-based live-accumulation and transformation of log data to arbitrary formats (MXML, XES, CSV).

Semantic log purging Based on expert interviews, we identify a set of fundamental constraints that a process has to obey. These constraints are used to semantically clean the log data obtained from the transformation step. In particular, the constraints help to enforce certain expected behaviors in the mined models. If it is possible to identify parallel branches, the constraints can also be used to separate these branches, with the intention to mine them separately.

2.2 Raw Data from a Blended Learning Environment

In order to evaluate our approach we utilized data collected through our university's SOA learning platform [7]. In particular, we selected a set of 10 different

computer science courses (process) that were known to (1) be conducted every year, (2) have a reasonably stable underlying process, and (3) utilize one or more of 4 learning services (Forum, Submission and Grading, Registration, Program Code Evaluation). For the purpose of this paper, we selected one course, which took place 3 times over the last three years, was attended by over 330 students (some of them attended multiple times) and yielded a total of 18511 events.

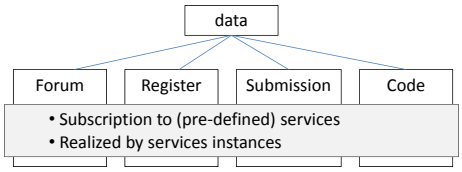

Fig. 2. Abstract View on HEP Data

As depicted in Fig. 2, the data was collected from a set of existing learning services. The particular course we selected consisted of Forum (to ask questions), Registration (for appointments and topics) and Submission (for uploading exercises). For this paper, we created a snapshot and anonymized the original data.

2.3 The Reference Process Model

From interviews with the involved actors in selected learning process, we created the following reference process model (cf. Fig. 3 and Fig. 4) in order to be able to compare it to the results of the mining. The model describes the process as envisioned by the three course instructors. For each student (process instance) the course starts with a `Kickoff Meeting`, followed by the parallel execution of three subprocesses. Within `Exercises` the students have to solve up to 7 tasks consisting of different subtasks within a certain time frame (cf. Fig. 4). After the timely `Upload` of a solution, the student `presents` the solution. This presentation is `evaluated` by the instructor. In parallel to all subtasks, the students might ask questions in the `Forum`. A similar procedure is executed for the `Milestone` subprocess during which students upload and present solutions that are evaluated by the instructor. For both, `Exercises` and `Milestones` the student collects points. This process, though quite small, offers possible challenges for process mining, i.e., parallelism, ad hoc process fragments, and loops.

3 Transformation of Temporal Application Data to Process Logs

As described in the previous section, in the real world logs often are not readily available but have to be aggregated from a multitude of data sources and formats. Sometimes, processes include applications that, in turn, allow for a series

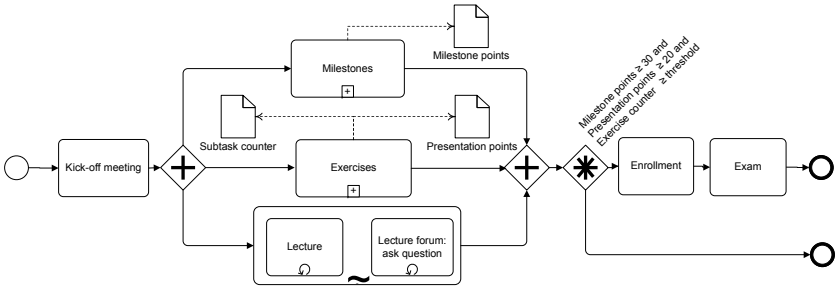

Fig. 3. Reference Course Process Model (Super Process in BPMN Notation)

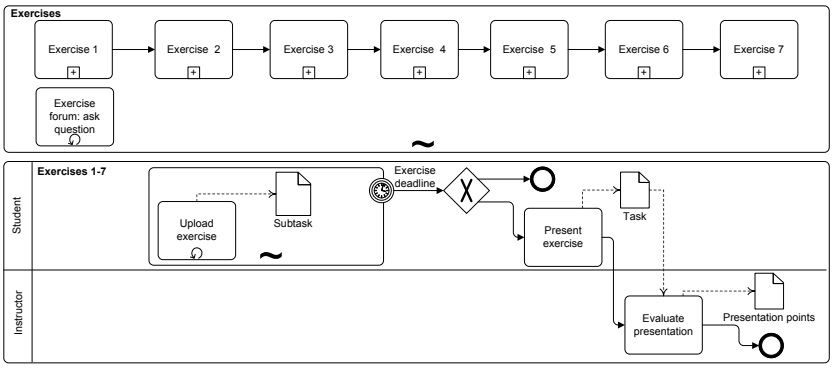

Fig. 4. Reference Course Process Model (Subprocess `Exercises` in BPMN Notation)

of steps (i.e. subprocesses). In order to incorporate such information into process logs, often a series of complex transformation steps involving databases has to be carried out (which cannot be done at runtime). The goal is to generate chronological log files suitable for process mining.

In order to avoid the above mentioned shortcomings, we designed a data extraction and transformation method building on a functional query-based domain specific language (DSL) written in Haskell. The full tool-chain used for extracting the data for this paper is available under `https://github.com/indygemma/uni_bi2`. Advantages of our approach include:

- Access data from heterogeneous sources without manual intermediate steps.
- Utilization of data streams instead of tables. This leads to efficient memory utilization for large data sets.
- For the data streams, arbitrary nesting of selections, projections, updates, and joins becomes possible.
- Once defined, transformations can be reused for equal or similar data sources.

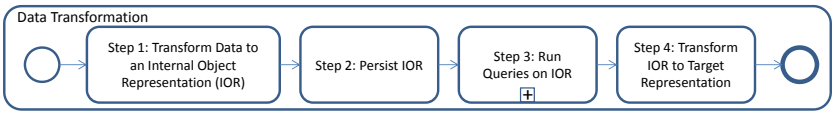

Fig. 5. The Data Transformation Subprocess

In the following, we give an overview of the involved concepts. We choose a CSV based format as it can be imported into process mining and semantic log purging tools. The CSV file consisted of: (1) timestamp, (2) instance id (i.e student id), (3) agent id (i.e originator of the event), (4) role (i.e. student, lecturer, tutor), (5) task name, and (6) data elements encoded as JSON (Javascript Object Notation). We use the four-step process depicted in Fig. 5 to break up transformation tasks for single data sources into a series of subtasks.

Step 1 - Transform to IOR: In order for queries to work uniformly on heterogeneous data, we transform the original data sets into an internal object representation (IOR). Data formats that are relevant for our example are: CSV, XML and raw file metadata (access time, file size). Since some of the data is deeply hierarchical, one of the design goals was to keep the hierarchical nature intact in the IOR. In order to also enable data stream operations, children are enriched with data properties of their parents (push down). For our data, an object instance corresponds to a single XML/CSV element (e.g. student). The IOR is thus populated by crawling once over the initial data set and creating object instances for each encountered element.

Step 2 - Persist IOR: In step 2, the IOR is compressed and serialized to disk. For step 3, files are only read when required. The resulting list of files represents the index on which queries can operate in the next step.

Step 3 - Run Queries on IOR: In this step, we define query functions that operate on the previously defined object streams. The idea is to apply arbitrary transformations on the objects before extracting them and passing them on for conversion to the target representation. The index is iterated until all the objects required for the subsequent step are present. The basic available operations are derived from relational algebra: select and join. Furthermore, it is possible to update (transform) objects according to custom rules. All operations return object streams, thus allowing for arbitrary nesting of operations. The DSL further supports a *Fuzzy Join*, which works on value ranges or value functions. The final operation in step 3 typically is the data extraction, that transforms objects into data rows. Fig. 6 shows a simple example, where a person record and a group record are selected, then joined (after the group records hierarchical structure is enriched (update)) and finally extracted. The extraction function required in our case was *groupSortAll*. Altogether, for our course data set, the algorithm as depicted in Fig. 7 was applied.

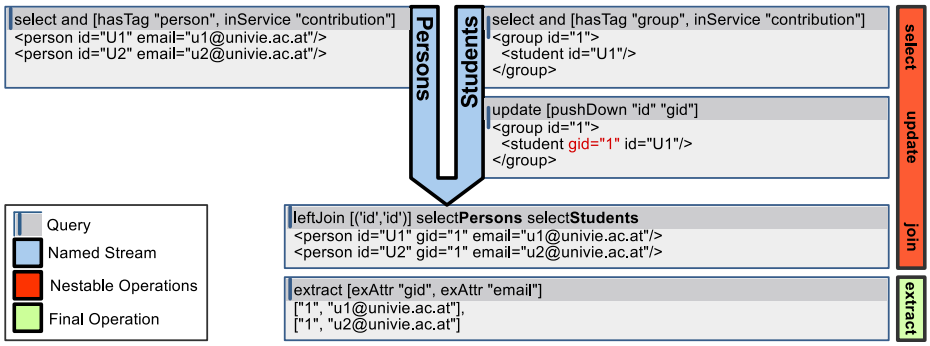

Fig. 6. A Simple Query Example

Step 4 - Transform IOR to Target Representation: In step 4, the extracted object rows are transformed into the final log (format).

A related tool for data transformation is XESame [6], which enables domain experts without programming experience to create transaction logs from data sources. This is achieved by a GUI-driven interface, wherein the user defines the mapping of source fields to target fields in the log. To make this mapping possible, XESame expects the input data to be tabular. Our approach does not make assumptions on the input nor on the output data, allowing all kinds of structured data formats – be they hierarchical or tabular – to serve as input. Employing an embedded DSL does require users to be comfortable with programming; but this allows for arbitrary complex data transformations in the form of flexible queries applied uniformly on heterogeneously structured data.

4 Semantic Log Purging

In contrast to existing algorithmic approaches to deal with noise data such as the Heuristics Miner [3], semantic log purging aims at cleaning the log at the semantic level. The basic idea of semantic log purging is to improve the quality of the logs by checking the encoded cases for compliance with fundamental domain-specific constraints. The logs representing cases that are incorrect with respect to these constraints are purged from the log set. It is notable that while noise stemming from spurious data (e.g., missing events) is often characterized by its

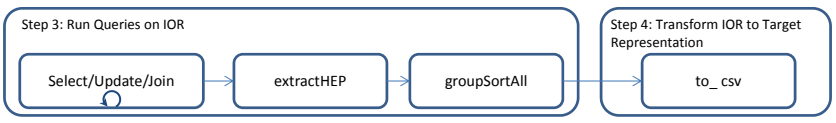

Fig. 7. Algorithm Applied Within Step 3

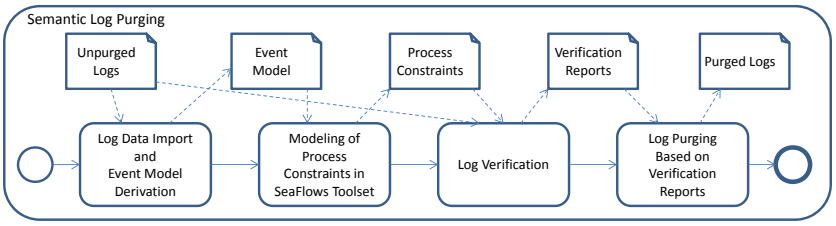

Fig. 8. The Overall Process of Semantic Log Purging

low frequency of occurrence within the logs, cases violating semantic constraints can even outnumber the cases complying with the constraints if this behavior is enabled by the system. Thus, semantic log purging may favor less frequent behaviors (e.g., *infrequent traces* as described in Sect. 1) and introduces a bias with respect to desired properties of the mined process model. Unlike frequency-based log cleaning, our approach does not purge infrequent traces unless they violate imposed constraints.

The specific semantic constraints relevant to the application can be obtained for example by interviewing domain experts. Clearly, the choice of semantic constraints used for log purging heavily affects the mined process models. Therefore, experimenting with constraints with different enforcement levels (e.g., high or low) can be helpful to identify the constraint set that leads to the best results with respect to the desired process model properties described in Sect. 2.1. Thus, an iterative approach is recommended.

The overall process of semantic log purging is illustrated in Fig. 8. The particular steps are detailed in the following.

Step 1 - Log import: We opted for the SeaFlows Toolset [8] in order to conduct semantic log purging. SeaFlows Toolset is a framework for verifying process models, process instances, and process logs against imposed constraints. It enables graphical modeling of constraints based on the events and their parameters (e.g., originator or event-specific data) contained in the logs. In an initial step, the logs in the form of CSV files are imported into SeaFlows Toolset where an event model for the logs is established. In particular, all events and their parameters are identified and can be used for specifying constraints.

Step 2 - Constraint specification: Based on interviews of a domain expert, we identify fundamental semantic constraints that have to be obeyed by the cases. Table 1 summarizes these constraints. We modeled the identified constraints in SeaFlows using the event model of the process logs. Fig. 9 shows a screenshot of the SeaFlows graphical constraint editor where the constraints are designed. In SeaFlows, a constraint is modeled as an directed graph consisting of a condition part and consequence parts. Fig. 10 depicts the modeled constraint c_2 and the corresponding logic formula. More details on the underlying formalism called *compliance rule graphs* is provided in [9].

Table 1. Examples of Semantic Constraints Imposed on the Example Course

	Constraint	Enforcement level
c_1	For each milestone, no upload must take place after the corresponding milestone deadline.	high
c_2	For each exercise, no upload must take place after the corresponding exercise deadline.	high
c_3	For each uploaded milestone, the instructor gives feedback.	low

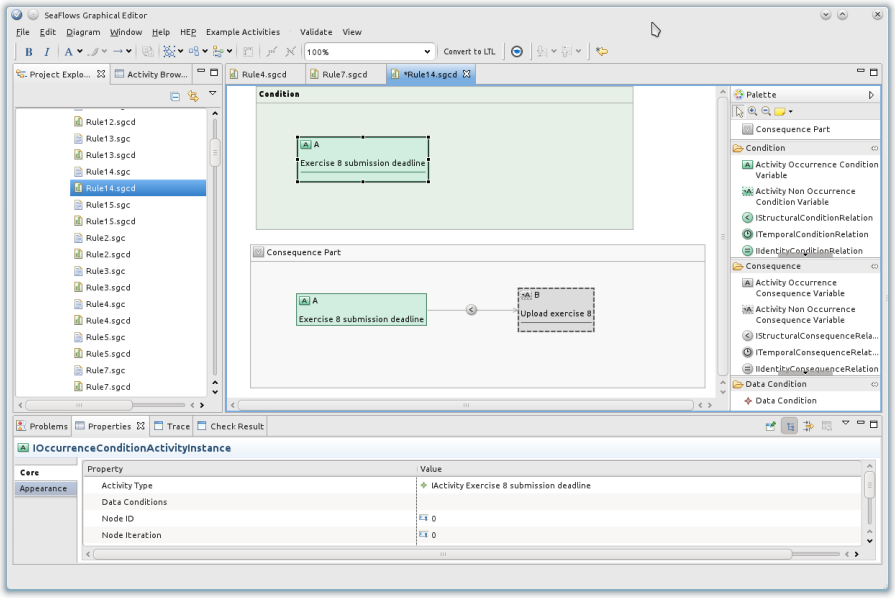

Fig. 9. Screenshot of the SeaFlows Graphical Constraint Editor

Step 3 - Log verification: In the third step, the cases of the log are automatically checked for compliance with the modeled constraints. In order to select specific branches for mining, there need to be additional constraints to remove tasks from unwanted branches. In our example we created separate logs for exercises and milestones. As shown in Fig. 11, SeaFlows Toolset then provides a detailed report on the detected violations of each case. In particular, for each case the violation report details which constraints are violated through which events.

Step 4- Log purging: Based on the detailed verification reports, we purged the log. For the case study, we removed only those cases violating constraints with strict enforcement level as only these really conflict with the expected process behavior. As illustrated by Fig. 1, the resulting log set is then used for the process mining (i.e. the Heuristics Miner which is resilient towards noise [3]).

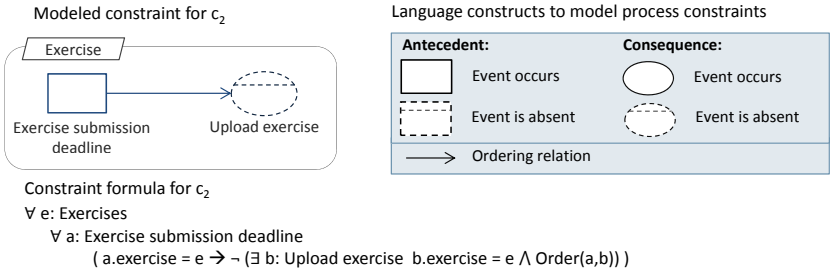

Fig. 10. Example of Constraint c_2

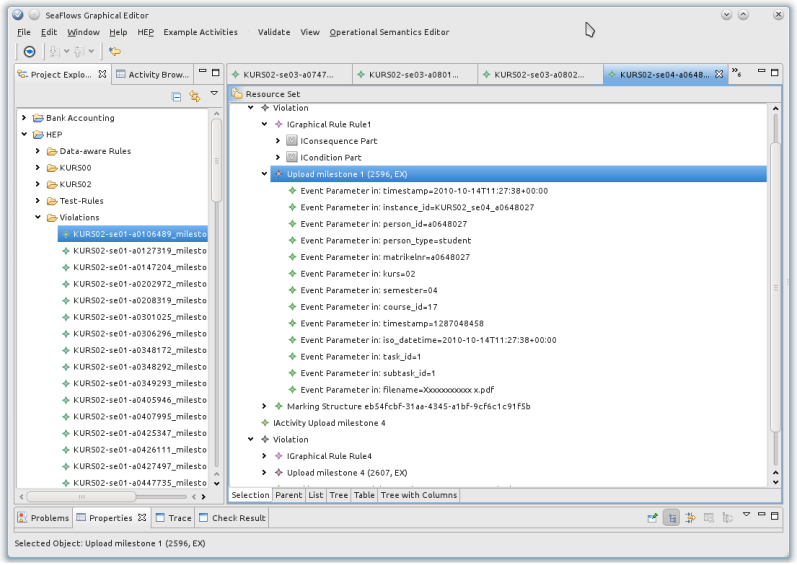

Fig. 11. Screenshot of the Violation View

5 Evaluation Based on HEP

As discussed in Section 2.1 and illustrated in Fig. 1, we compare the process models mined from the original and from the purged logs against the expected reference models. For the overall case study, we analyzed ten course types (among them a course on advanced database systems (DBS) that serves as example in the paper). The example course has 18511 events and more than 330 cases. For brevity, we will present details on only the example course in the following.

5.1 Observations and Findings

Table 2 compares the process models mined from the original and from the purged log with each other. As discussed in Sect. 4, we exploited semantic knowledge about the subprocesses to mine them separately as their parallel nature makes it very difficult to mine meaningful process models otherwise. For each semester, in which the course took place, we mined a separate process model, respectively, as the particular process may vary in each semester.

Table 2. Quantitative Comparison of the Mined Process Models

	Process Model: Original		Process Model: Purged	
Subprocess	Tasks	Edges	Tasks	Edges
Exercise (SE01)	21	31	21	30
Milestone (SE01)	20	22	20	26
Exercise (SE03)	29	38	29	36
Milestone (SE03)	27	37	27	34
Exercise (SE04)	29	40	26	34
Milestone (SE04)	23	32	23	32

Qualitative analysis revealed two cases: the process models mined from purged logs are equal or smaller (case **A**)/ bigger (case **B**) than the original models (w.r.t. the nodes and edges). Interestingly, all models mined from purged logs are closer to the expected reference models (cf. Section 2.3) than the models mined from the original logs. In case **A**, spurious edges were removed in the models from purged logs reducing the "spaghetti-degree" of the models. In case **B**, the original models were overabstracted. Here, the models mined from purged logs contain more expected edges. In the `Exercise` subprocess (SE04), optional activities were removed from the model. In the `Milestone` subprocess (SE04), a spurious edge was replaced by an expected edge.

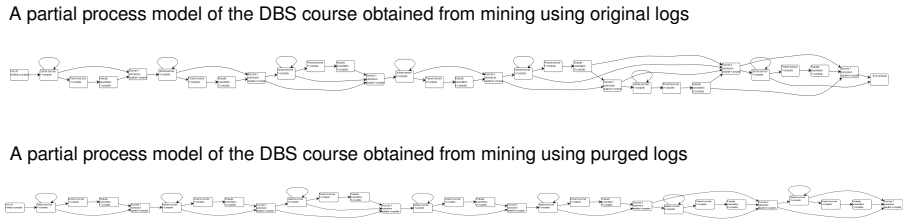

Fig. 12. Original vs. Purged Logs (`Exercise` Subprocess)

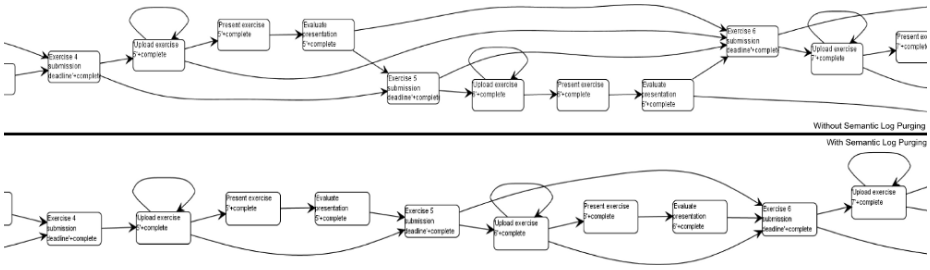

Fig. 13. Original Mined Process Model vs. Process Model Mined from Purged Logs

Example Fig. 12 depicts the Exercise subprocess (SE03) mined from the original process logs and the one mined from purged logs. The process models cover the activities of the example course concerning the exercises conducted in the course. As the figure shows, both models mainly differ in the last part where the originally mined process model is more complex (i.e., has more edges) than the process model mined from the purged log.

In the detail view of these processes in Fig. 13, we can see that the model mined from the original logs contain edges that are not allowed by the reference process model. For example, from the submission deadline of Exercise 4 there is a path to the submission deadline of Exercise 6 without passing the submission deadline of Exercise 5. As the deadlines are system generated events, this cannot occur in practice. Such paths in the mined process model are caused by the submission events occurring after the submission deadline, which violates the imposed constraints (cf. Table 1). In contrast, the process model mined from the purged logs does not contain such edges as the cases with submission events after the deadline are removed from the log set.

5.2 Lessons Learned

We observed that semantic log purging indeed improves the quality of the mined models with respect to the properties described in Sect. 2.1. We observed that with a small set of constraints the process models mined from the purged logs are already quite close to the reference models. Due to the semantic log filtering, ad-hoc changed instances that violate the constraints were purged from the data set (cf. case **(4)** in Sect. 1) and thus do not contribute to the mined models.

As we also used constraints to extract branches (for separate mining), it was interesting to see that after removing all unnecessary events and incorrect / incomplete cases (noise), often only a small fraction of cases remained (see Fig. 14). The separate mining of parallel branches has proven to be a valid approach to reduce the "spaghetti-degree" of process models mined from log data contributed by parallel branches

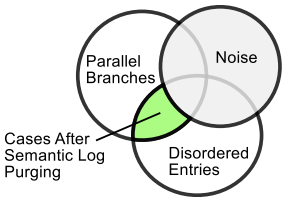

Fig. 14. Fraction of Log Entries Used for Mining

(cf. case **(2)** in Sect. 1). As the approach does not rely
on frequency, infrequent traces are not filtered out and
thus, can be incorporated in the mined process models (cf. case **(3)** in Sect. 1).

Altogether, semantic log purging is an approach to clean the log with respect to expected properties and thus to support the mining of reference models from the process logs. It introduces a bias with respect to the resulting process model. Therefore, it "guides" the mining process.

It should be noted that the collection of process constraints can happen independently of whether or not expected reference models are created before the mining process. We also experimented with bigger sets of constraints containing also constraints with a lower enforcement level (nice-to-have constraints). As most logs violate these constraints, the data set would have become too small. Therefore, a lesson learned is that an iterative approach (consisting of choosing constraints, log purging, mining, re-choosing constraints, ...) is helpful to determine a suitable set of constraints. We can also consider the automatic application of constraint subsets in order to find an optimal set.

6 Related Work

In general, data quality is a crucial issue for applying analysis techniques such as data or process mining. For preparing data accordingly different cleaning techniques have been developed during the last decades, cf. [10]. In particular, for cleaning data from errors or impossible values, integrity constraints have been utilized in the database area [11]. Though the basic idea is similar to the one of this paper, there are two basic differences. First of all, no process-oriented data (represented by process logs) has been subject to data cleaning approaches so far. Process-oriented data might be particularly complex, considering at least control and data flow, time-relations, and organizational assignments. Accordingly, the associated process constraints might be quite complex as well, e.g., regulatory packages such as BASEL III. Further on, integrity constraints are mostly independent of the application context, hence correcting data errors can be accomplished without further domain knowledge (e.g., extinguishing data with AGE $<$ 0 [11]. In this paper, we extend these application-independent techniques by a data cleaning approach that utilizes knowledge on the application context of the data, representing, for example, knowledge on the order of certain teaching courses. In practice, the existence of such knowledge can often be assumed, for example, medical guidelines [12], internal quality controls, or contractual policies [13].

Process mining offers a multitude of techniques to analyze logs from past process executions [1]. By now, a variety of tools for process analysis is comprised by the process mining framework ProM [6]. Key to the effectiveness of process mining is the acquisition and (pre-)processing of suitable logs. Here, particular challenges arise from the various sources of log data in real-world applications. In [14], Funk et al. describe their setup for product usage observation by means of process mining. Their approach allows for semantically annotating

the logged data using ontologies. Mans et al. report on their experience from a case study on the application of process mining in the hospital context in [15]. To receive intelligible models, they had to abstract from low-level events. In our case, we rather had to provide more low-level events in order to obtain meaningful process models. As data sources are heterogeneous, no general approach can be provided for log pre-processing besides some fundamental strategies such as filtering particular events.

A common strategy to deal with "spaghetti-models" is to abstract from infrequent behavior (considered as noise) to yield simpler models, for example employed by the Heuristics Miner [3]. Here, we would like to stress that our approach does not rely on the frequency of observed behavior but rather introduces expectations with respect to the mined model. In [4], Fahland et al. introduce an approach to structurally simplify a mined process model while preserving certain equivalence notions. Such approaches are orthogonal to our work.

For auditing process logs with respect to certain properties (e.g., the constraints for semantic log purging), ProM offers the LTL Checker [16], a tool that allows for checking properties specified in *linear temporal logic* (LTL). The original LTL Checker only works at the granularity of activitiy labels. Thus, additional data conditions as used by some of the constraints we experimented with are not directly supported. An extension of the LTL Checker is introduced in [17] in the context of *semantic process mining* that allows for using concepts from ontologies as parameters of an LTL formula. Application of this approach requires the establishment of an ontology.

7 Summary and Outlook

This paper provides new techniques for data transformation and cleaning embedded within a case study to show the feasibility of the approach. In particular, we introduced a query-based data transformation approach that is able to transform temporal application data into process logs in different target representations (e.g., CSV). We further propose semantic log purging as an approach to improve the quality and intelligibility of the mined process model. In contrast to algorithmic approaches that deal with noise data by for example applying heuristics and thus can only capture incorrect process executions occurring infrequently, our approach cleans the log with respect to expected properties. In the case study within the HEP project, we were able to confirm the feasibility of our approaches. In future work, we will additionally examine the information on constraint-violating cases to analyze the reasons for deviations from the process.

References

1. van der Aalst, W., et al.: Process mining manifesto. In: Business Process Management Workshops. LNBIP (2011)
2. De Medeiros, A.K.A., Weijters, A.J.M.M.: Genetic process mining: an experimental evaluation. Data Mining and Knowledge Discovery **14** (2007)

3. Weijters, A., van der Aalst, W.: Rediscovering workflow models from event-based data using little thumb. ICAE **10** (2003) 151–162
4. Fahland, D., van der Aalst, W.M.P.: Simplifying mined process models: An approach based on unfoldings. In: Proc. BPM 2011. Volume 6896 of LNCS., Springer (2011) 362–378
5. van der Aalst, W.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, Berlin Heidelberg (2011)
6. Verbeek, H.M.W., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: XES, XESame, and ProM 6. In: Information Systems Evolution - CAiSE Forum 2010. Volume 72 of LNBIP., Springer (2010) 60–75
7. Derntl, M., Mangler, J.: Web services for blended learning patterns. In: Proc. IEEE International Conference on Advanced Learning Technologies. (2004) 614–618
8. Ly, L.T., Knuplesch, D., Rinderle-Ma, S., Göser, K., Pfeifer, H., Reichert, M., Dadam, P.: SeaFlows Toolset - Compliance verification made easy for process-aware information systems. In: Information Systems Evolution - CAiSE Forum 2010. Volume 72 of LNBIP., Springer (2010) 76–91
9. Ly, L.T., Rinderle-Ma, S., Dadam, P.: Design and verification of instantiable compliance rule graphs in process-aware information systems. In: Int'l Conf. on Advanced Information Systems Engineering. (2010) 9–23
10. Rahm, E., Do, H.: Data cleaning: Problems and current approaches. IEEE Data Engineering Bulletin **23** (2000) 313
11. Heiko Müller, J.F.: Problems, methods, and challenges in comprehensive data cleansing. Technical Report 164, Humboldt University Berlin (2003)
12. Dunkl, R., Fröschl, K.A., Grossmann, W., Rinderle-Ma, S.: Assessing medical treatment compliance based on formal process modeling. In: Information Quality in e-Health. Volume 7058. Springer Berlin Heidelberg (2011) 533–546
13. Rinderle-Ma, S., Mangler, J.: Integration of process constraints from heterogeneous sources in Process-Aware information systems. In: Int'l Workshop Enterprise Modelling and Information Systems Architectures - EMISA 2011. (2011)
14. Funk, M., Rozinat, A., de Medeiros, A.K.A., van der Putten, P., Corporaal, H., van der Aalst, W.M.P.: Improving product usage monitoring and analysis with semantic concepts. In: Proc. UNISCON 2009. LNBIP, Springer (2009) 190–201
15. Mans, R.S., Schonenberg, H., Song, M., van der Aalst, W.M.P., Bakker, P.J.M.: Application of process mining in healthcare - a case study in a dutch hospital. In: BIOSTEC (Selected Papers). Volume 25 of CCIS., Springer (2008) 425–438
16. van der Aalst, W., de Beer, H., van Dongen, B.: Process mining and verification of properties: An approach based on temporal logic. In: Proc. OTM Conferences 05. Volume 3761 of LNCS. (2005) 130–147
17. de Medeiros, A.K.A., van der Aalst, W.M.P., Pedrinaci, C.: Semantic process mining tools: Core building blocks. In: Proc. ECIS 2008. (2008) 1953–1964